SOFTWARE MAINTENANCE: RESEARCH AND PRACTICE, VOL. 8, 269–280 (1996)

Practice

Evolutionary Method of Delivery as Applied to a Large Re-engineering Effort

SHANTHA MOHAN
Consilium Inc., 485 Clyde Avenue, Mountain View, CA 94043 U.S.A.

SUMMARY

Since its introduction in 1982, Consilium's flagship software product has gone through several major and minor revisions. The product, WorkStream, is a manufacturing execution system used mostly in the fast-moving semiconductor and pharmaceutical industries to reduce cycle times, reduce costs and improve yields. In the early 1990s, Consilium faced the problem of continuing to satisfy the needs of its current customers while at the same time, working towards meeting the requirements of its future prospects.

This paper presents Consilium's experience in re-engineering WorkStream in the context of the evolutionary method of delivery and the phase-in and phase-out conversion methods of re-engineering. Consilium emphasized three of the key elements of the evolutionary method of delivery: (1) multiple objectives, (2) open architecture for growth, and (3) early delivery of components that are of highest value to cost ratio. The paper concludes with a summary of the benefits derived from this experience and the lessons learned.

KEY WORDS: evolutionary delivery; open architecture; phase–in conversion; phase–out conversion; value to cost ratio; manufacturing control systems; software maintenance

1. INTRODUCTION

The evolutionary method of delivery (EMD) is based on the principles of delivering measurable value-added deliverables to the real end–user (customer), of giving priority to system components having the highest value to cost ratio, and of adjusting the design of the final deliverables based on feedback (Gilb, 1988, pages 83–113). Planning for an EMD involves setting multiple objectives, being user (customer) and results driven, and building an open–ended architecture. Some of the benefits expected from using the EMD are:

- It enables us to see the true costs of the system at an earlier stage, thus making it possible to do any course correction needed to achieve final objectives.
- The final design is likely to meet user needs more closely.
- Users get something that is usable early in the development cycle and hence are likely to support development activities more enthusiastically.

CCC 1040–550X/96/040269–12

Received 7 September 1995
Revised 22 March 1996

- The estimating process keeps getting refined, resulting in successive deliveries being more successful with respect to cost and delivery dates.
- Developers working on the effort can see the fruits of their labour and will be further motivated to achieve more.

Also documented in the literature are the importance of software evolution due to the complexity of today's software systems (Van Horn, 1980; Belady, 1979; Belady and Lehman, 1976; Basili and Turner, 1975), as well as the growing complexity of user requirements. An important point to note is that it is difficult to build a system that addresses all of the users' requirements and get it right in the first attempt. Frederick Brooks's paper on software engineering (Brooks, 1987) talks about the concept of *growing* and not *building* software. While the paper is about prototypes, the ideas apply equally to re-engineering efforts on huge systems.

This paper reports on the experience at Consilium in re-engineering applied to its flagship software product, WorkStream. WorkStream is a leading product in the class of manufacturing execution systems (MES) used widely in the semiconductor and pharmaceutical industries to improve yields in manufacturing processes, to reduce cycle times, and reduce costs. Because the semiconductor and pharmaceutical industries are dynamic and competition-sensitive, the market for WorkStream offers continual opportunities for Consilium to make modifications in its flagship product.

This paper describes Consilium's experience in using the EMD in re-engineering its product WorkStream, in combination with two of the three conversion methods. The three usual methods are the big-bang (not used by Consilium), the phase-in, and the phase-out (Feiler, 1993).

This paper is organized as follows: Section 2 describes the situation and problems at Consilium. Section 3 describes the multiple objectives that needed to be achieved with the re-engineering effort. Section 4 describes the rationale behind the re-engineering methodology used. Section 5 describes the open architecture that became the framework for the re-engineering effort and ties it back to the multiple objectives. Section 6 presents the development staffing for the effort. Section 7 presents the current status of the effort. Section 8 summarizes the lessons learned. Finally section 9 presents the conclusions.

2. BACKGROUND

A manufacturing execution system (MES) is a software application that monitors and controls manufacturing operations as they occur on the plant floor. Consilium's flagship product WorkStream was conceived in the early 1980's and has been in use for over ten years.

WorkStream is a comprehensive software system consisting of many modules that users (customers) can selectively implement depending on requirements (see Figure 1). The core modules consist of 'work in process (WIP) tracking' that tracks lots or batches going through the production process, and 'resource tracking' that tracks resources used in production such as equipment and operators. All the other modules are built around these core modules. The original system runs on VAX/VMS, and uses a CODASYL database. The user interface is character-cell based, full screen, and menu driven. The language of implementation is COBOL. Over the years the software was enhanced to support more modules. It also acquired an X window/Motif-based user interface.

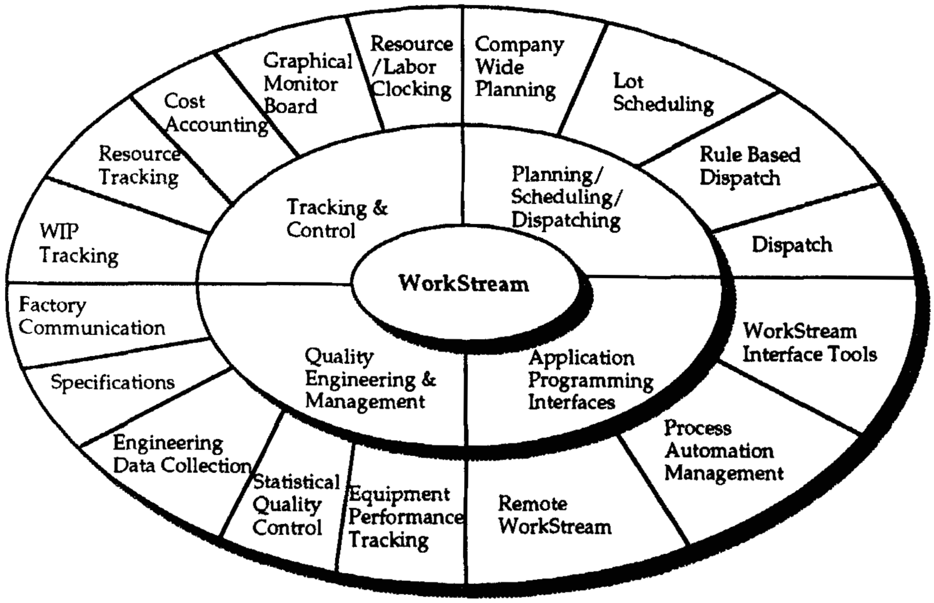

Figure 1. WorkStream is a comprehensive manufacturing execution system

The number of lines of source code in the entire system is approximately four million. The bulk of the code belongs to work in process tracking followed by resource tracking. The system is monolithic in nature. It has one large executable for the on–line system that encompasses the monitor and control functions. Also, it has several smaller executables that can be run in batch mode to generate reports and extracts. A number of compute–intensive functions such as scheduling and archiving are also part of the batch executables. The system design is primarily transaction–based. For example, users who want to effect a change in an equipment's status would execute a WorkStream transaction from one of the items under the resource tracking menu. The transaction validates the users' input, ensures data integrity and records the change in the WorkStream database.

The cross module functions are integrated primarily through the application code and the use of the database. WorkStream accepts more than 2 000 kinds of transactions. The tight integration of modules implies that every time new functionality was to be added, the changes to the system were extensive.

As the 1990s approached it was clear that this system needed to be upgraded to meet the technological demands of the market place. Sales prospects wanted open systems, relational databases and graphical user interfaces. In an effort to address these needs of the market place, WorkStream was ported to run on Unix using a relational database system instead of a CODASYL system. This move to Unix gave the product greater acceptance in the market. The training required to accomplish the porting eased the transition of the software development team to Unix and relational database.

While the move to Unix was considered a good intermediate step, it did not address issues of maintainability of the monolithic system such as:

- *Cost.* The cost of using and maintaining a monolithic system is felt in several areas.

The users cannot distribute the load onto multiple machines for fault-tolerant operation. The monolithic system also requires significant downtime for any upgrades which lower the throughput of the plant. Significant investment in initial hardware costs must be made for future growth.
- *Quality*. Bug fixes to the code have a rippling effect, thus affecting the quality of the software. Every fix to the software required a considerable amount of testing, not only for Consilium, but also the user (customer) who might have customized the software.
- *Customer service*. Requests for enhancements to the software take time to implement because of the amount of code that needs to be handled. This reduces the quality of service.

The industry trend dictated a client/server type of solution. Consilium decided to investigate developing a fully distributed manufacturing execution system to address these needs. With the expansion in the semiconductor industry, an opportunity to implement a new, state-of-the-art system for that industry presented itself.

3. OBJECTIVES

Many of WorkStream's current users have been using the product since the early 1980s. Some of these users have customized the software in many ways including interfacing with other systems such as material requirements planning (MRP) and manufacturing automation. WorkStream software had more than adequate functionality for running their plants as paperless facilities. Even automated operations have found value in WorkStream. When the cost of system downtime (in lost production) is hundreds of thousands of dollars per hour, the users tend to be very risk-averse. Users wanted the software to continue to run their plants, but expected incremental improvements that will increase ease of use. The same users however wanted to see a next generation system for their new plants and, additionally, wanted a safe migration path for their old implementations of WorkStream.

Consilium faced the problem of continuing to satisfy the needs of its current users while at the same time working towards meeting the requirements of its future prospects. The development of Consilium's next generation system named WorkStream DFS (distributed factory system) was the answer to this problem. The problem of satisfying the installed base while addressing the market demands is not new. The process Consilium followed to address this issue is one that can help other software providers transition between generations of complex systems.

Based on the key concepts of planning for evolutionary method of delivery, the multiple objectives for the WorkStream DFS project are as follows:

- Provide a state-of-the-art manufacturing execution system to the semiconductor and electronic industry.
- Construct this system in such a way that will help existing customers migrate to the new architecture with minimum risk and downtime.
    - Leverage the investment in the existing system, both in hardware and internal support staff skills as well as existing data and custom programs.
    - Phase-in the new applications with new technologies at a rate that is manageable.

- Provide this solution in incremental steps of highest value to the users.
- Leverage the application knowledge base within Consilium's software group.
- Transition Consilium's COBOL personnel to become proficient in state-of-the-art object-orientated technology, thereby providing a career growth path for them.

4. RE-ENGINEERING METHODOLOGY

Three approaches to renovating software are the big-bang approach, phase-out approach and the phase-in approach (Feiler, 1993). In the big-bang approach, the new system is built from scratch, though some parts of the legacy system might be recycled. Once the new system is ready, the old system is put out of commission, and fully replaced by the new system. In the phase-out or incremental development approach, while the existing legacy system is still in production, new components built on the desired architecture are added to the existing legacy system, replacing some of the legacy system capabilities. The phase-in or evolutionary development approach introduces new functionality to the legacy system, slowly evolving the system into the target system. These three approaches are variants of the traditional systems changeover approaches respectively of immediate replacement, parallel operations and phased installation (see any of the usual texts, for example, Jordan and Machesky, 1989, pages 580–584).

For WorkStream DFS, after an initial false start, the big-bang approach was ruled out because of the following reasons:

- The risks associated with installing (in billion dollar plants) a brand new system that has not been proven elsewhere were too high.
- The effort required to execute the big-bang approach was more than 200 person-years, and such a magnitude in itself posed a big risk in terms of project size.
- With a big-bang approach, there was no way to deliver a small installment of functionality that could actually be tried in a plant environment to validate the product direction.

A combination of the phase-in and phase-out approaches was chosen to implement the WorkStream DFS project. In using these approaches, careful consideration was given to how the system was to be decomposed into modules (Parnas, 1972), and the importance of designing systems for ease of extension (Parnas, 1979).

System capabilities that did not exist in WorkStream, but could deliver big benefits to the users as additions to WorkStream were considered for the phase-in approach. As noted earlier, this is a key EMD principle. An example of a phase-in component of WorkStream DFS is the 'recipe management system'. It provided an ideal phase-in component of the new architecture because it promised the highest ratio of value delivered to development cost. In semiconductor processing, the use of a wrong recipe could lead to yield busts and the loss of hundreds of thousands of dollars. The requirements of a recipe management system in preventing incorrect processing were well understood and could be featured in a first release of the system to provide enough benefits to the user. As the requirements became more clear, the system could be extended to phase in additional features.

System capabilities of WorkStream, which provided limited benefits to the users were chosen for the phase-out approach. The 'data collection' and 'statistical quality control'

capabilities of WorkStream (Figure 1) lacked a graphical user interface and was a drawback to using the capability. By providing improved functionality along with ease of use in configuration and graphical user interface (GUI) capabilities, substantial benefits could be delivered in a DFS component called 'quality server'. The design of the system allows users to continue using data collection and statistical quality control capabilities of WorkStream, while incrementally implementing the capabilities of the quality server in DFS. Eventually the phase–out method will be heavily used in replacing the 'work in process' and 'resource tracking' modules of WorkStream.

5. WORKSTREAM DFS ARCHITECTURE

One aspect of planning for EMD is to design systems with extensible architecture. The architecture chosen for WorkStream DFS is based on the use of a message bus. The system is composed of atomic application servers that collaborate with each other using the message bus as the communication mechanism. By developing applications with well-defined message interfaces, additional applications can be added to the configuration as the users' needs become known. The functionality of the WorkStream software is made available through an object adapter layered on the message bus. In this way, we could use the core functionality of WorkStream, while continuing to add new functionality or replace existing functionality with additional servers, eventually replacing the core functionality itself.

RDO/ISIS (reliable distributed objects from ISIS distributed systems) was chosen as the message bus for implementing DFS. Choosing a commercially available message bus allowed Consilium to focus on its core competency of building applications instead of building and maintaining the communication mechanism that formed the basis of the architecture. The fault–tolerance and load–balancing aspects of the commercial message bus chosen were also important to the design of the system. Figure 2 shows the evolution of the WorkStream DFS architecture.

The object–orientated components will be built using different languages. Smalltalk was chosen for the implementation of the first servers for a number of reasons. Portability across a variety of platforms, a very productive development environment, and a graphical user interface builder were some of the reasons. Future servers will be built in Smalltalk or C++. The goal of the DFS architecture was to allow applications written in any language to communicate with each other over the message bus. Figure 3 shows the first application components of WorkStream DFS. Implementation of a communication layer on the message bus was the first step in the evolution of the architecture. This layer allows Consilium to replace the chosen message bus with any other bus with minimal effort. The applications would require no change.

Connecting WorkStream to the bus was made possible by the application programming interface (API) modules in the legacy system, which are the process automation module and the remote WorkStream. These modules communicated with external systems through the VMS mail box. Now they communicate with the message bus through an object adapter. Without these API modules, making WorkStream part of the distributed architecture would have been very difficult.

Other DFS framework services evolved with the phase–in application components. Security server provides services to the DFS modules for validating the users and enforcing

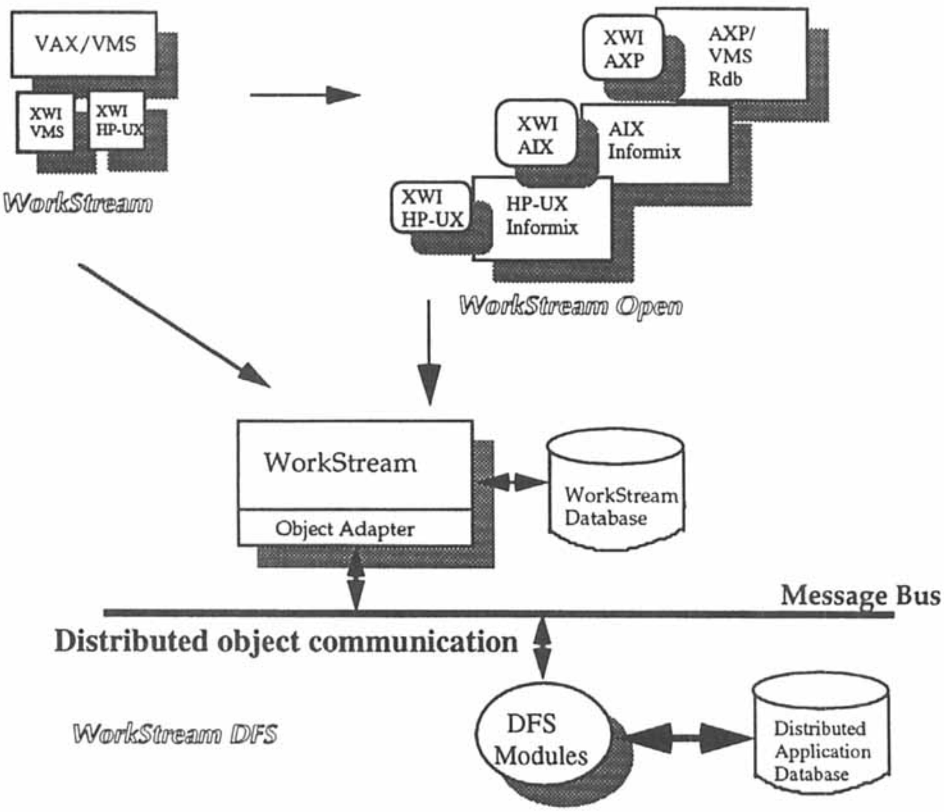

Figure 2. The evolution of WorkStream DFS

the privileges allowed for different authority levels of users. DFS manager provides services to monitor and manage the distributed applications.

The message bus provides the basis for fault-tolerant operation. Using the concept of process groups, the DFS framework implements fault tolerant and load balancing behaviours. An RDO/ISIS capability called the co-ordinator–cohort mode of operation allows multiple processes of the same application to belong to a process group, and facilitates the distribution of process load as needed. Another RDO/ISIS capability called the primary backup mode of operation guarantees having at least one process available at all times to process requests.

6. STAFF—APPLICATION EXPERIENCE REUSE AND UPGRADE

Prior to starting the WorkStream DFS project, the WorkStream software group was made up of personnel with a skill set that included mostly COBOL, some C, and a little assembly programming. The personnel were proficient in database technology usage and knowledgeable about the application. Through various cross-department programs, they had access to the end-user and an understanding of the real business problems being solved. With the porting of WorkStream to Unix, the developers had become familiar with that environment as well.

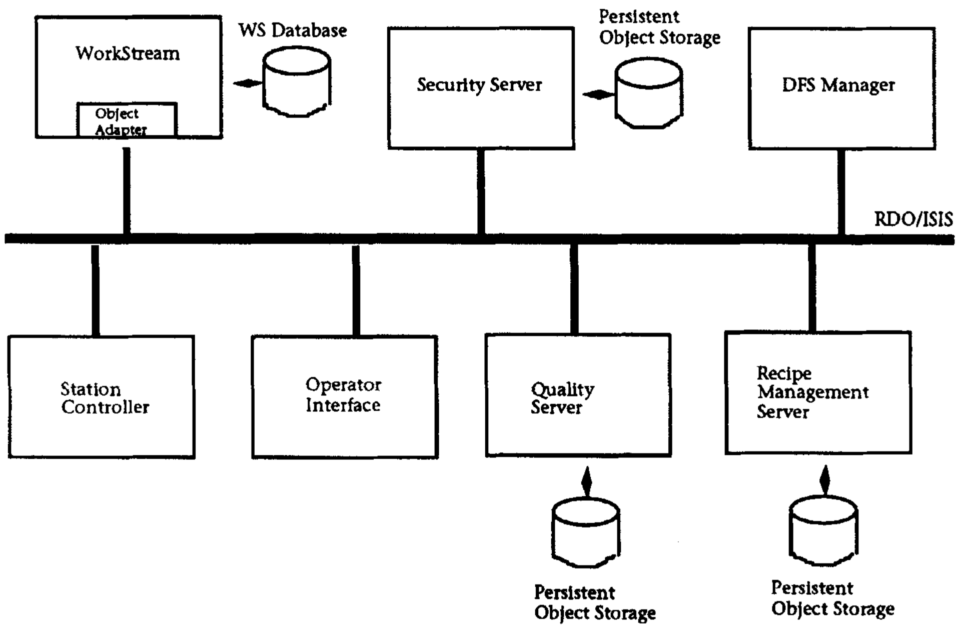

Figure 3. The first WorkStream DFS components

At the start of the DFS project, Consilium identified a team to work on some prototypes using the new technology. This team went through training classes on Smalltalk and object–orientated methodology. The staff chosen to participate had a good knowledge of the application and were able to participate in the analysis of the application problem and propose solutions.

As the prototype was being designed, groups of developers were given Smalltalk training and slowly phased into the project. A number of object–oriented development experts and Smalltalk programmers were brought in to mentor the team and helped in rapid prototyping exercises. Over the last two years, we have managed to reuse the application experience we have accumulated in-house for the past several years. At the same time, we have also substantially raised the technical skills of the team.

7. LESSONS LEARNED

The current status of the WorkStream DFS product validates the benefits of the EMD for large re-engineering efforts such as on Consilium's WorkStream project:

- The phase–in module recipe management system provides huge benefits with respect to the cost of developing that capability. This has made the users of the that system enthusiastic about considering additional capabilities in other areas.
- The project forced the issue of handling the legacy system interface early on which presented some difficult problems. In solving these problems, we have become quite experienced in building and modifying distributed systems whose components may be written in multiple languages. This has helped us build a robust framework.
- The EMD allows us to make use of new technology standards as they become

available. For example, we are in a good position to make use of standards such as the OMG's CORBA.
- The software personnel in the project are very motivated in their efforts because of the feedback they get from the users of the first components.
- Consilium's installed customer base is starting to see the viability of continuing to use WorkStream and of eventually migrating to a fully distributed MES such as WorkStream DFS.

The lessons learned so far from this re-engineering effort are in two areas, business and technical. Let us look first at the business area.

- The most valuable business lesson learned is that re-engineering efforts have to be tailored to the business you are in. The characteristics of the market serviced by WorkStream are that it requires a high reliability, high availability solution, and that it has limited expansion potential. That is, additional business comes mostly from the installed customer base.
- A second business lesson learned is that re-engineering the existing legacy system was more cost effective than developing a new replacement system. The cost or effort required to produce a system that is functionally equivalent to, or better than the legacy system was prohibitive in the case of WorkStream, and did not fit in the window of opportunity.
- A third business lesson learned is that application expertise must be conserved. Consilium had to find a way to keep the existing staff applied to the support of the legacy product, while bringing out the re-engineered product.
- A fourth business lesson we learned early was that the big-bang approach was not feasible given our goals. To keep the installed customer base, we had to evolve the product technically.
- A fifth business lesson learned is the importance of listening to the installed customer base. If the existing users do not see a path forward from the legacy system, they do not support your efforts, and question the maintenance dollars they spend on the product. With the EMD, the customers did not raise that question adversely.

Some of the lessons learned under the technical category are these:

- The first technical lesson learned is that the language (Smalltalk) chosen to implement the system provided a very good introduction to object-orientated programming for the COBOL staff. It raised the technical capabilities of most of the staff, and gave them a positive feeling in terms of the technical skills needed in today's market place. It also worked against us—we lost several members of the staff who could now go out and command very high salaries elsewhere.
- The second technical lesson learned is that the cost of testing a distributed system appears to be enormous compared to a monolithic system. The number of failure scenarios in a distributed system is high. Accounting for them and providing a comprehensive test suite is not a task to be taken lightly.
- The third technical lesson learned is that more rigorous approaches to regression and performance testing are required for a distributed system. Regression testing a distributed system multiplies the normal difficulties faced in regression testing.

- The fourth technical lesson learned is that transitioning to maintaining a distributed system involves continued learning of new skills because of the complex interactions involved in developing, deploying and maintaining distributed systems.

8. STATUS

The re-engineering efforts for WorkStream DFS continue. Additional distributed modules of WorkStream DFS are under development. In 1996, Consilium will introduce another high benefit application called 'alarms and alerts', which in its first release will provide the capability to manage by exception. Efforts are also under way for providing the equivalent of the core of WorkStream—the work in process tracking and the resource tracking capabilities—in WorkStream DFS, which is a major technical challenge. Table 1 summarizes the progress of the re-engineering effort.

By 1998, the expectation is to have a fully distributed system that replaces the core functionality of WorkStream. This is diagrammed in Figure 4. We plan to execute this conversion effort of replacing the core of WorkStream by focusing maintenance work in the following categories:

- Extract and document current core functionality.
- Identify additional requirements of today.
- Design target systems.
- Map existing functionality to a subset of target capabilities.
- Provide wrappers to legacy capability to function in parallel with target functionality so as to provide migration path for existing installations.
- Develop and deploy target core functionality.

Table 1. Progress of the WorkStream re-engineering effort

Year	Component released
1993	Initial version of WorkStream connection Initial version of station controller
1994	Enhancements to WorkStream connection Enhancements to station controller Initial version of recipe management server
1995	Enhancements to WorkStream connection Enhancements to station controller Enhancements to recipe management server Initial version of quality server Initial version of security server Initial version of operator interface
1996	Enhancements to WorkStream connection Enhancements to station controller Enhancements to recipe management server Enhancements to quality server Initial version of alarms and alerts server

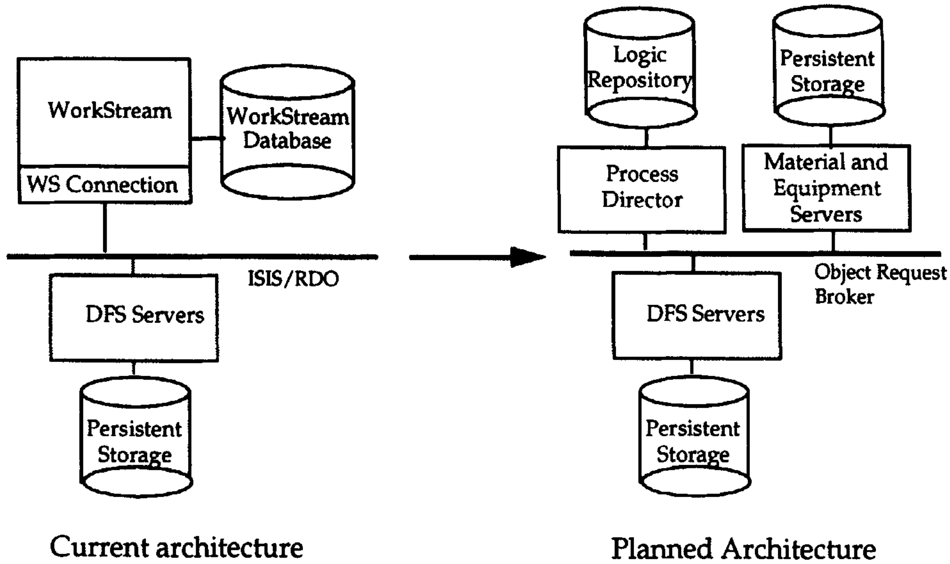

Figure 4. Planned architecture change for moving WorkStream to the fully distributed manufacturing execution system WorkStream DFS

9. CONCLUSIONS

For Consilium, a big-bang approach would not have met the existing customers' needs, nor deliver the functionality required by the market in sufficient time. An evolutionary approach has proven a superior solution for Consilium to the problem faced by all organizations having legacy systems—how to deliver a next generation system to an installed customer base with minimal disruption and risk.

The experience from Consilium's WorkStream project validates the benefits of the EMD (evolutionary method of delivery). Applying its principles in conjunction with practicing the phase-in and phase-out of components has allowed Consilium and its customers to see some significant early benefits, while giving a solid foundation to continue the re-engineering efforts. Consilium's positive experience in using EMD on its four million line WorkStream system has given us the confidence to continue moving ahead with EMD towards our objectives of providing benefits to our customers, as well as our company.

Acknowledgements

The author acknowledges the contributions of the WorkStream DFS project team members and the valuable review comments from Dr. Jonathan Golovin, Chief Technology Officer and Chairman of Consilium Inc. The author thanks the anonymous reviewers for their helpful guidance.

References

Basili, V. R. (1990) 'Viewing maintenance as reuse-oriented software development', *IEEE Software*, **7**(1), 19–25.

Basili, V. R. and Turner, A. J. (1975) 'Iterative enhancement: a practical technique for software development', *IEEE Transactions on Software Engineering*, **SE-1**(6), 390–396.

Belady, L. A. (1979) 'Evolved software for the 80's', *Computer*, **12**(2), 79–82.

Belady, L. A. and Lehman, M. M. (1976) 'A model of large program development', *IBM System Journal,* **15**(3), 225–252.
Brooks, F. P. (1987) 'No silver bullet: essence and accidents of software engineering', *Computer,* **20**(4), 10–19.
Feiler, P. H. (1993) 'Re-engineering: an engineering problem', Technical Report CMU/SEI-93-SR-5, Software Engineering Institute, Pittsburgh PA.
Gilb, T. (1988) *Principles of Software Engineering Management,* Finzi, S. (Ed), Addison–Wesley Publishing Company, Reading, MA.
Jordan, E. W. and Machesky, J. J. (1989) *Systems Development,* PWS–Kent Publishing Co., Boston, MA.
Parnas, D. L. (1972) 'On the criteria to be used in decomposing systems into modules', *Communications of the ACM,* **15**(12), 1053–1058.
Parnas, D. L. (1979) 'Designing software for ease of extension and contraction', *IEEE Transactions on Software Engineering,* **SE–5**(2), 128–138.
Van Horn, E. C. (1980) 'Software must evolve', in Freeman, H. and Lewis, P. M. (Eds), *Software Engineering,* Academic Press, Inc., New York, NY, pp. 209–226.

Author's biography:

Shantha Mohan is a director of software development in Consilium Inc., a company based in Mountain View, California, where she manages the WorkStream DFS product group. In her eleven years of experience in manufacturing execution systems, she has developed and managed different application areas of WorkStream, and is now managing the project to re-engineer WorkStream into WorkStream DFS. She has a Ph.D. in Operations Management from Carnegie Mellon University and a bachelor's degree in Electronics and Communication from Madras University, India.